Guaranteed recursive nonlinear state estimation using interval analysis

Michel Kieffer, Luc Jaulin, Eric Walter

Proceedings of the 37th IEEE
Conference on Decision & Control
Tampa, Florida USA • December 1998

Guaranteed Recursive Nonlinear State Estimation Using Interval Analysis

Michel KIEFFER*, Luc JAULIN+ and Éric WALTER*[1]
*Laboratoire des Signaux et Systèmes, CNRS-Supélec
Plateau de Moulon, 91192 Gif-sur-Yvette, France
{kieffer, walter}@lss.supelec.fr
+Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers
2 bd Lavoisier, 49045 Angers, France
jaulin@babinet.univ-angers.fr

Abstract

The problem considered is state estimation in the presence of unknown state and measurement noise, each noise component being assumed to belong to some known interval. In such a bounded-error context, most available results are for linear models, and the purpose of the present paper is to deal with the nonlinear case. Based on interval analysis and the notion of set inversion, a new state estimator is presented, which evaluates a set estimate *guaranteed* to contain all values of the state that are consistent with the available observations, given the noise bounds and a set containing the initial value of the state. To the best of our knowledge, it is the first time that such a guaranteed estimator is made available. The precision of the set estimate can be improved, at the cost of more computation. The theoretical properties of the estimator are studied, and computer implementation has received special attention. A simple illustrative example is treated.

Keywords: bounded errors, interval analysis, nonlinear estimation, set estimation, state estimation.

1 Introduction

This paper is concerned with recursive nonlinear state estimation in the context of bounded state noise and measurement noise. Unknown and possibly time-varying parameters can also be considered. The problem is to characterize the set of all state vectors of a given model that are compatible with the measured outputs and assumed error bounds. Usually, the model is assumed to be linear and ellipsoids are computed at each step, guaranteed to contain the present state [13], [14], [10], [4] and [5]. Computation closely parallels Kalman filtering, alternating prediction and correction phases. For nonlinear models, the problem is much more difficult, since the set is usually nonconvex and may even be nonconnected. Our purpose here is to characterize such a set recursively, thus facilitating real-time implementation. This characterization will be performed with the help of interval analysis, by computing outer approximations by unions of boxes, or *subpaving*, with arbitrary precision. Interval techniques have already been applied to building an Interval Kalman Filter [3] for uncertain linear models, but no guarantee was given that the state would belong to the evaluated set. In contrast, the technique to be presented here applies to nonlinear and possibly uncertain models, and guarantees its results.

In Section 2, an idealized state estimator will be presented. After a few introductory words on an illustrative example, basic interval ideas will be recalled in Section 3. They will then be used to build an approximate but guaranteed nonlinear state estimator using subpavings. The correction and prediction steps of this estimator will be presented in Sections 4 and 5 respectively. Section 6 summarizes the algorithm and applies it to the illustrative example.

2 State estimation

Consider the nonlinear and possibly time-varying system defined by

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k), \\ \mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \mathbf{w}_k, \end{cases} \quad k = 0, 1, \ldots \qquad (1)$$

where $\mathbf{u}_k \in \mathbb{R}^m, \mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{y}_k \in \mathbb{R}^p$ are respectively the input, state, and output vectors. The initial state $\mathbf{x}_0$ is assumed to belong to some prior compact set $\mathcal{X}_0 \subset \mathbb{R}^n$. $\{\mathbf{v}_k\}$ and $\{\mathbf{w}_k\}$ are unknown state and measurement noise sequences, assumed to belong to the known intervals sequences $\{[\mathbf{v}]_k\}$ and $\{[\mathbf{w}]_k\}$. $\mathbf{f}_k$ and $\mathbf{h}_k$ are known functions (or finite algorithms).

[1] Corresponding author

The problem to be considered is the recursive estimation of the smallest set $\mathcal{X}_l$ guaranteed to contain all values of $\mathbf{x}_l$ compatible with the information available at time l, *i.e.*, with $\mathcal{I}_l = \left\{\mathcal{X}_0, \{\mathbf{u}_k, \mathbf{y}_k, [\mathbf{v}]_k, [\mathbf{w}]_k\}_{k=0}^{l}\right\}$. Note that the more general problem of joint state and parameter estimation can easily be treated in this context.

Let $\mathcal{X}_l$ be a set containing all values of $\mathbf{x}_l$ compatible with $\mathcal{I}_l$ and $\mathcal{X}_{l+}$ be the set of all values of the state that are reachable from some $\mathbf{x}_l$ in $\mathcal{X}_l$ with input $\mathbf{u}_l$ and state noise $\mathbf{v}_l \in [\mathbf{v}]_l$

$$\mathcal{X}_{l+} = \mathbf{f}_l\left(\mathcal{X}_l, \mathbf{u}_l, [\mathbf{v}]_l\right) = \left\{\mathbf{f}_l\left(\mathbf{x}, \mathbf{u}_l, \mathbf{v}_l\right) | \mathbf{x} \in \mathcal{X}_l, \mathbf{v}_l \in [\mathbf{v}]_l\right\}.$$

Moreover, let $\mathcal{Y}_{l+1}$ be the set of all admissible values of the output, when its measured value is $\mathbf{y}_{l+1}$

$$\mathcal{Y}_{l+1} = \mathbf{y}_{l+1} - [\mathbf{w}_{l+1}] = \left\{\mathbf{y}_{l+1} - \mathbf{w}_{l+1} | \mathbf{w}_{l+1} \in [\mathbf{w}]_{l+1}\right\},$$

and let $\mathcal{X}_{l+1}^{o}$ be the set of all values of $\mathbf{x}$ which could have led to an observation $\mathbf{y} \in \mathcal{Y}_{l+1}$

$$\mathcal{X}_{l+1}^{o} = \mathbf{h}_{l+1}^{-1}\left(\mathcal{Y}_{l+1}\right) = \left\{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}_{l+1}\left(\mathbf{x}\right) \in \mathcal{Y}_{l+1}\right\}.$$

A set containing all values of $\mathbf{x}_l$ compatible with $\mathcal{I}_{l+1}$ is then given by $\mathcal{X}_{l+1} = \mathcal{X}_{l+} \cap \mathcal{X}_{l+1}^{o}$. As $\mathcal{X}_0$ contains all possible values of the initial state vector $\mathbf{x}_0$, one may thus, at least theoretically, recursively evaluate a set $\mathcal{X}_l$ containing all possible values of the state vector at each step l. These ideas are summarized in the following *idealized algorithm*:

For $l = 0$ to L, do

1. Prediction: $\mathcal{X}_{l+} = \mathbf{f}_l\left(\mathcal{X}_l, \mathbf{u}_l, [\mathbf{v}]_l\right)$.
2. Correction: $\mathcal{X}_{l+1} = \mathbf{h}_{l+1}^{-1}\left(\mathcal{Y}_{l+1}\right) \cap \mathcal{X}_{l+}$.

Proposition 2.1 *$\mathcal{X}_l$, as computed by the idealized algorithm, is the smallest set guaranteed to contain $\mathbf{x}_l$ that can be computed from $\mathcal{I}_l$.*

Proof: See [9]. ∎

Except in very particular cases, it is not possible to evaluate the sets $\mathcal{X}_{l+}$, $\mathcal{X}_{l+1}^{o}$ and $\mathcal{X}_{l+1}$ exactly. Our purpose will therefore be to get a guaranteed outer approximation of $\mathcal{X}_{l+1}$, as accurate as possible. Before presenting the basic blocks of an algorithm for this task, a simple illustrative example will be introduced.

Bouncing-ball example: Consider an ideal ball of radius r bouncing on the floor. The motion is assumed to be one-dimensional, and the state of the ball is characterized by its height x_1 and speed $x_2 = \dot{x}_1$. Our purpose is to evaluate the state $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$ of the ball dropped with an initial state $\mathbf{x}_0$ only known to belong to $\mathcal{X}_0$ from noisy discrete-time measurements of its height, according to $y_k = h_k\left(\mathbf{x}_k\right) + w_k = [1\ 0]\mathbf{x}_k + w_k$, with w_k belonging to some known interval. Since no friction is considered, and the ball is assumed incompressible, the motion is straightforwardly described by $(\dot{x}_1 = x_2, \dot{x}_2 = -g)$ during the fall, and by $(x_1 \to x_1, x_2 \to -x_2)$ when the ball hits the floor. By exact discretization, a finite algorithm $\mathbf{f}$ evaluating $\mathbf{x}_{k+1}$ for a given $\mathbf{x}_k$ is easily constructed.

3 Intervals and subpavings

The prediction and correction steps of the idealized algorithm can be implemented in an approximate but guaranteed way using interval computation and subpavings, which provide a powerful tool for the description of sets. The aim of this section is to recall the basic notions of interval arithmetic that are necessary to build subpavings. For a more detailed presentation of interval arithmetic, see, *e.g.*, [11] or [6]. A scalar interval $[x]$ is a closed, connected and bounded subset of $\mathbb{R}$. It may be characterized by its lower bound $\underline{x}$ and upper bound $\overline{x}$. Thus, $[x] = \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}$. An important characteristic of $[x]$ is its *width* $w\left([x]\right) = \overline{x} - \underline{x}$. An n-dimensional *box* (or vector interval) $[\mathbf{x}]$ is the Cartesian product of n scalar intervals $[x_i]$. The width of $[\mathbf{x}]$ is $w\left([\mathbf{x}]\right) = \max_{i=1,\ldots,n} w\left([x_i]\right)$.

Interval arithmetic provides an extension to intervals of the usual arithmetical operations on reals $\{+, -, \times, /\}$ through the generic formula $\circ \in \{+, -, \times, /\}$, $[x] \circ [y] = \{x \circ y | x \in [x] \text{ and } y \in [y]\}$. All these operations are *inclusion monotonic* [11]: if $[x'] \subset [x]$ and $[y'] \subset [y]$, then $[x'] \circ [y'] \subset [x] \circ [y]$. This means that uncertainty on their results cannot deteriorate if the uncertainty on their arguments is improved, which suggests algorithms where intervals are split into subintervals to increase precision. Extension to vector and matrix intervals, necessary for state estimation, can be found in [11] and [6].

Let f be a real function of a real variable, defined on $\mathcal{D} \subset \mathbb{R}$. The definition of this function is extended to intervals $[x] \subset \mathcal{D}$ as follows: $f\left([x]\right) = \{f\left(x\right) | x \in [x]\}$. It is straightforward to extend any monotonous real function to interval arguments. Interval counterparts to simple non-monotonic functions such as sine or cosine are less trivial to obtain, but exact image intervals can still be computed by simple algorithms [6]. In general, however, it is not possible exactly to characterize the image set, which is not even necessarily an interval. The fundamental notion of an inclusion function makes it possible to compute *an interval* guaranteed to contain this image set. An *inclusion function* associated with f will be denoted by $f_{[]}$; for any $[x] \subset \mathcal{D}$, it should satisfy $f\left([x]\right) \subseteq f_{[]}\left([x]\right)$. In addition to being in-

clusion monotonic, it is desirable that $f_{[]}([x])$ converge to $f([x])$ when $w([x])$ tends to zero. When $f_{[]}([x])$ possesses this property, it is said to be *convergent*. Among the infinitely many inclusion functions associated with any given real function f, it is of interest to get one that is as accurate as possible. Various techniques are available to obtain efficient inclusion functions, especially when the width of $[x]$ is small enough [11].

3.1 Describing sets using subpavings

An n-dimensional *subpaving* $\widehat{\mathcal{X}}$ is a union of non-overlapping boxes of $\mathbb{R}^n$. The set of all subpavings of $\mathbb{R}^n$ is denoted by $\mathcal{K}(\mathbb{R}^n)$. To characterize the precision with which a subpaving may approximate a compact subset of $\mathbb{R}^n$, it is necessary to introduce a distance between such subsets. Let $\mathcal{A}$ and $\mathcal{B}$ be compact subsets of $\mathbb{R}^n$ and $\mathbf{x}$ be a vector of $\mathbb{R}^n$. We shall use the *Hausdorff distance* based on the infinity norm: $d(\mathcal{A},\mathcal{B}) = \max\{d_0(\mathcal{A},\mathcal{B}), d_0(\mathcal{B},\mathcal{A})\}$, where $d_0(\mathcal{A},\mathcal{B}) = \max_{\mathbf{a}\in\mathcal{A}} d_0(\mathbf{a},\mathcal{B})$, with $d_0(\mathbf{x},\mathcal{B}) = \min_{\mathbf{b}\in\mathcal{B}} d_\infty(\mathbf{x},\mathbf{b})$ and $d_\infty(\mathbf{x},\mathbf{b}) = \max_{i=1,\ldots,n}\{|x_i - b_i|\}$. *Hausdorff distances* such as $d(.,.)$ are metrics for the set of compact subsets of $\mathbb{R}^n$ [2]. Any compact subset of $\mathbb{R}^n$ can be represented with any desired precision using subpavings. (see [9] for more details). As subpavings consist of boxes, it is now possible to extend interval operations to subpavings. The way subpavings will be described will have a direct impact on the complexity of these operations.

To illustrate how subpavings will be represented, consider the two-dimensional example of Figure 1. The compact set $\mathcal{X}$ (in black), included in the box $[\mathbf{x}_0] = [0,8]^2$, is represented by the subpaving $\widehat{\mathcal{X}}$ (in grey).

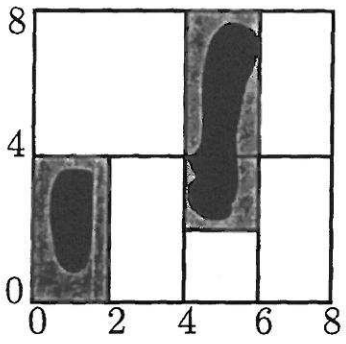

Figure 1: Set $\mathcal{X}$ (in black) described using the subpaving $\widehat{\mathcal{X}}$ (in grey).

The subboxes of $\widehat{\mathcal{X}}$, as described by Figure 1, can be enumerated in a list

$$\mathcal{L} = \{[0,2]\times[0,4], [4,6]\times[2,4], [4,6]\times[4,8]\}.$$

As all these boxes are obtained from $[\mathbf{x}_0]$ by a succession of bisections and selections, one may alternatively store the initial box together with the history of these operations. It is then necessary to introduce a notation to describe how boxes are bisected and selected. If $[\mathbf{x}]$ is an n-dimensional box and if $m\left([x]_j\right) = \left(\overline{x_j} + \underline{x_j}\right)/2$ denotes the midpoint of $[x]_j$, then

$$L_j[\mathbf{x}] = \left([x]_1, \ldots, [x]_{j-1}, \left[\underline{x_j}, m\left([x]_j\right)\right], [x]_{j+1}, \ldots, [x]_n\right)$$
$$R_j[\mathbf{x}] = \left([x]_1, \ldots, [x]_{j-1}, \left[m\left([x]_j\right), \overline{x_j}\right], [x]_{j+1}, \ldots, [x]_n\right)$$

will respectively be called the *left* and *right subboxes* resulting from the bisection of $[\mathbf{x}]$ across its jth dimension. Recursively using the way boxes are bisected and selected, one may write the list $\mathcal{L}$ as $\mathcal{L} = \{L_1L_2L_1[\mathbf{x}_0], R_2L_1L_2R_1[\mathbf{x}_0], L_1R_2R_1[\mathbf{x}_0]\}$. This description is obviously not unique. For instance, $R_1L_2L_1[\mathbf{x}_0] = L_2R_1L_1[\mathbf{x}_0]$. This ambiguity can be avoided by agreeing on some canonical bisection rule. The canonical rule used in this paper is to cut every box $[\mathbf{x}] \subset \mathbb{R}^n$ across its *main dimension*, defined as $j = \inf\{i = 1,\ldots,n \mid w([x]_i) = w([\mathbf{x}])\}$, but other canonical rules are easy to construct.

A subpaving will be called *regular* if it can be obtained by applying the canonical rule. For regular subpavings, the cut direction can easily be retrieved, so indexing L and R is no longer necessary and $\mathcal{L}$ can be written as $\{LLL[\mathbf{x}_0], RLLR[\mathbf{x}_0], LRR[\mathbf{x}_0]\}$.

Given that most interval algorithms include a bisection process dividing a *parent* box into two *children*, it is of interest to consider $\widehat{\mathcal{X}}$ as an *ordered binary-tree* [1], *i.e.*, a finite set of *nodes*, which either is empty or consists of one node, the *root* of the tree, and two disjoint ordered binary trees, the *left* and *right subtrees*. For $\widehat{\mathcal{X}}$ of Figure 1, this binary tree is represented by Figure 2.

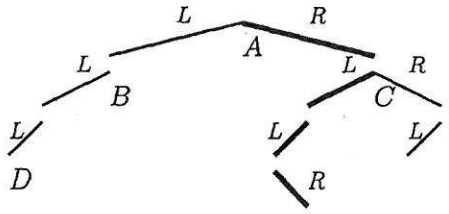

Figure 2: Tree representation of $\widehat{\mathcal{X}}$ of Figure 1.

A is the *root* of the tree. B and C are respectively the *left* and *right children* of A: they are *siblings* because they have the same *parent* A. A has *left* and *right subtrees*; B has an *empty* right subtree, both are *nodes*, as they have at least one non-empty subtree. At last, D is a *leaf*, because it has two empty subtrees. This tree is constructed using the list $\mathcal{L}$ of subboxes. The growth of the branches is determined by the way in which the initial box $[\mathbf{x}_0]$, corresponding to the root, is bisected. A node indicates that the associated

subbox has been bisected according to the canonical bisection rule. A leaf indicates that the corresponding subbox is entirely in the subpaving. For example, the bold branch of Figure 2 corresponds to the subbox $RLLR[\mathbf{x}_0] = [4,6] \times [2,4]$. The *depth* of a subbox corresponds to the number of bisections applied to get this subbox from the root box. Thus, the depth of the box $[4,6] \times [2,4]$ is 4. A tree (subpaving) is said to be *minimal* if it has no siblings leafs (subboxes). Such siblings leafs should be eliminated, with their parent node becoming a leaf instead. The notions of binary trees and regular subpavings are equivalent, so the vocabulary for trees will be used for subpavings. In what follows, the tree representation will be adopted due to the natural recursivity of this data structure. But the equivalence between the tree and its list of boxes has to be kept in mind.

3.2 Implementing subpavings

All functions and algorithms that will be presented have been developed in the C-XSC framework [6].

A subpaving will be represented by a C++ class, whose private members are the classical tree components.

```
class spaving {
   ivector box;  // box represented by node or leaf
   spaving_ptr lchild;    // pointer to left subtree
   spaving_ptr rchild;};// pointer to right subtree
```

`lchild` (resp. `rchild`) points to the left (resp. right) subtree. A `NULL` pointer corresponds to an empty subtree, thus leafs are identified by `NULL` pointers `lchild` and `rchild`. The function `_box(spaving)` gives access to the box represented by the node; `_lchild(spaving)` and `_rchild(spaving)` respectively return the left and right child subtree.

The headers of some functions manipulating subpavings and used in the algorithms described in this paper are presented in Appendix A. More details can be found in [9].

The implementable counterpart of the idealized algorithm also alternates prediction and correction steps. As the correction step is simpler, it will be presented first.

4 Correction

This step requires characterizing $\mathcal{X}_{k+1} = \{\mathbf{x} \in \mathcal{X}_{k+} \mid \mathbf{h}_{k+1}(\mathbf{x}) \in \mathcal{Y}_{k+1}\}$. This task belongs to the class of *set-inversion* problems, formulated as follows: given two sets $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{Y} \subset \mathbb{R}^m$ and a function $\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m$, characterize the set $\mathbf{f}_{\mathcal{X}}^{-1}(\mathcal{Y}) = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{f}(\mathbf{x}) \in \mathcal{Y}\}$.

This problem is solved in an approximated but guaranteed way using the SIVIA (Set Inversion Via Interval Analysis) algorithm developed by Jaulin and Walter [7], [8]. This algorithm characterizes $\mathbf{f}_{\mathcal{X}}^{-1}(\mathcal{Y})$ by bracketing it between inner and outer subpavings. Its convergence has been studied in [8]. Here, a recursive version, evaluating the outer subpaving only, but suitable for subpaving manipulation will be presented.

Assume that $\mathcal{X}$ and $\mathcal{Y}$ are respectively enclosed in the subpavings $\widehat{\mathcal{X}}$ and $\widehat{\mathcal{Y}}$. The aim of SIVIA is to enclose the solution set $\mathcal{S} = \mathbf{f}_{\widehat{\mathcal{X}}}^{-1}\left(\widehat{\mathcal{Y}}\right) = \left\{\mathbf{x} \in \widehat{\mathcal{X}} \mid \mathbf{f}(\mathbf{x}) \in \widehat{\mathcal{Y}}\right\}$, into a subpaving $\widehat{\mathcal{S}}$, with the help of an inclusion function $\mathbf{f}_{[]}$ of $\mathbf{f}$. $\widehat{\mathcal{S}}$ is thus an approximate but guaranteed solution set. For this purpose, a recursive structure will be used to scan all nodes of $\widehat{\mathcal{X}}$. Let $[\mathbf{x}]$ be the box corresponding to a given node N of $\widehat{\mathcal{X}}$. If $\mathbf{f}_{[]}([\mathbf{x}]) \subset \widehat{\mathcal{Y}}$, the subpaving stemming from N is included in the solution set $\mathcal{S}$ and is therefore stored in the subpaving $\widehat{\mathcal{S}}$. If $\mathbf{f}_{[]}([\mathbf{x}]) \cap \widehat{\mathcal{Y}} = \emptyset$, $[\mathbf{x}] \cap \mathcal{S} = \emptyset$. Therefore, N and the subpavings stemming from it can be discarded from further consideration. If the results of the two preceding tests are negative and if $w([\mathbf{x}]) > \epsilon$, the subtrees stemming from N have to be tested, else the box corresponding to N is considered small enough to be incorporated in $\widehat{\mathcal{S}}$. The positive real ϵ is chosen by the user and specifies the desired accuracy of the description of the set to be characterized.

The complete C++ code of this recursive version of SIVIA is in Appendix B. SIVIA($\widehat{\mathcal{X}}$, $\mathbf{f}_{[]}$, $\widehat{\mathcal{Y}}$, ϵ) returns a subpaving containing $\mathbf{f}_{\widehat{\mathcal{X}}}^{-1}\left(\widehat{\mathcal{Y}}\right)$.

5 Prediction

Computing $\mathcal{X}_{k+} = \{\mathbf{f}_k(\mathbf{x}, \mathbf{u}_k, \mathbf{v}_k) \mid \mathbf{x} \in \mathcal{X}_k, \mathbf{v}_k \in [\mathbf{v}]_k\}$ is a problem of direct image evaluation, which is at the core of interval arithmetic: given two compact sets $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{S}_0 \subset \mathbb{R}^m$ and a function $\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m$, characterize the set $\mathcal{S} \subset \mathcal{S}_0$ such that $\mathcal{S} = \{\mathbf{f}(\mathbf{x}) \in \mathcal{S}_0 \mid \mathbf{x} \in \mathcal{X}\}$. In the special case $m = 1$, it has been shown [11], [12] that the approximation could be made as precise as desired provided that the inclusion function $f_{[]}$ is Lipschitz. When $m > 1$, we shall now show that it is still possible to approximate the image set with arbitrary precision, under conditions that depend on whether $\mathbf{f}$ can be inverted.

When $\mathbf{f}$ is invertible, prediction can again be cast in the formalism of set inversion, as the problem of finding $\mathcal{S} = \{\mathbf{x} \in \mathcal{S}_0 \mid \mathbf{f}^{-1}(\mathbf{x}) \in \mathcal{X}\}$. The prior search set $\mathcal{S}_0$ should be taken large enough to be guaranteed to contain the set of interest. If $\widehat{\mathcal{S}_0}$ and $\widehat{\mathcal{X}}$ are subpavings enclosing $\mathcal{S}_0$ and $\mathcal{X}$, then $\mathcal{S}$ can be approximated by $\widehat{\mathcal{S}}_\epsilon$ =SIVIA($\widehat{\mathcal{S}_0}$, $\mathbf{f}^{-1}$, $\widehat{\mathcal{X}}$, ϵ); provided that a conver-

gent inclusion function is available for $\mathbf{f}^{-1}$. Assuming that $\mathbf{f}$ is invertible may seem rather strong, but in many physical cases inverting the dynamic only means inverting time, so the inverse dynamic is rather simple to obtain from the direct one.

When $\mathbf{f}$ is not invertible, a specific and computationally more demanding procedure is needed. The basic idea of the direct IMAGE SubPaving evaluation procedure (IMAGESP) is to describe the initial set $\mathcal{X}$ using a nonminimal subpaving consisting of p boxes $[\mathbf{x}]_i$ whose width is less than ϵ. Then IMAGESP evaluates the image of each of these p boxes using an inclusion function $\mathbf{f}_{[]}$ of $\mathbf{f}$ and stores these images in a list $\mathcal{L}_\epsilon$. One thus gets p *image* boxes, each of which contains the true image set of the associated initial box. The image set $\mathcal{S}$ is therefore included in the union of all of them. At last, IMAGESP merges all these image boxes into a subpaving to allow further processing.

The complete C++ code of IMAGESP is given in [9]. $\widehat{\mathcal{S}}_\epsilon$ = IMAGESP($[\mathbf{s}]_0$,$\mathbf{f}$, $\widehat{\mathcal{X}}$, ϵ) returns the image subpaving $\widehat{\mathcal{S}}_\epsilon$ of the subpaving $\widehat{\mathcal{X}}$ by the function $\mathbf{f}$, or rather the part of it that is included in some prior search box $[\mathbf{s}]_0$.

The quality of the estimated image depends of course on that of the inclusion function $\mathbf{f}_{[]}$ of $\mathbf{f}$ and on the precision parameter ϵ. The precision can, at least in principle, be made arbitrarely good (see [9] for more details).

An approximate but guaranteed version of the idealized algorithm can now be proposed.

6 Guaranteed state estimator

For $l = 0$ to L, do

1. Guaranteed prediction *If* $\mathbf{f}_l$ *is invertible, then*

   $$\widehat{\mathcal{X}_{l+}} = \text{SIVIA}(\widehat{\mathcal{S}}, \mathbf{f}_l^{-1}, \widehat{\mathcal{X}_l} \times \{\mathbf{u}_l\} \times [\mathbf{v}]_l, \epsilon);$$

   where $\widehat{\mathcal{S}} = \{[\mathbf{s}]\}$ *is the search subpaving, consisting of a possibly very large box in which all states are assumed to stay. Else*

   $$\widehat{\mathcal{X}_{l+}} = \text{IMAGESP}([\mathbf{s}], \mathbf{f}_l, \widehat{\mathcal{X}_l} \times \{\mathbf{u}_l\} \times [\mathbf{v}]_l, \epsilon);$$

2. Guaranteed correction. *From* $\widehat{\mathcal{X}_{l+}}$, *select all elements that are compatible with measurements at step* $l+1$

   $$\widehat{\mathcal{X}_{l+1}} = \text{SIVIA}(\widehat{\mathcal{X}_{l+}}, \mathbf{h}_l, \mathcal{Y}_{l+1}, \epsilon);$$

Bouncing ball example (continued): Sampling time was $T = 1$ s, $r = 0.2$ m and g was taken as 10 $m.s^{-2}$. The (unknown) true initial state was $\mathbf{x}_0 = (x_0, \dot{x}_0)^{\mathrm{T}} = (5, 0)^{\mathrm{T}}$. It was only assumed to belong to $[\mathbf{x}_0] = [3,6]\times[-3,3]$. During motion, state was assumed to stay within the box $[0,8] \times [-12,12]$. At each time $t = kT$, $k \in \mathbb{Z}_+^*$, the position of the ball was measured, with the measurement noise w_k assumed to belong to $[-0.2, 0.2]$. The observation equation was $y_k = x_k + w_k$. No state noise was considered. At each step l, the predicted subpaving $\widehat{\mathcal{X}_l^+}$ was evaluated using SIVIA, and is displayed on Figure 3 in light grey. The corrected subpaving $\widehat{\mathcal{X}_{l+1}}$ is in dark grey.

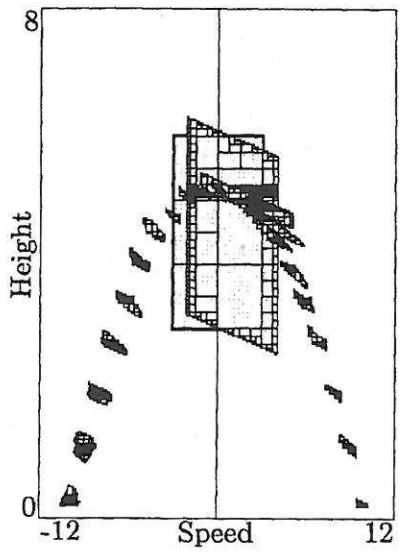

Figure 3: Ball motion. Initial state $\mathbf{x}_0 = (x_0, \dot{x}_0)^T$ is in $[3,6] \times [-3,3]$.

7 Conclusions

A new recursive nonlinear state estimator has been presented. At any given time, it returns a set guaranteed to enclose all values of the state that are consistent with the information available so far.

As in classical Kalman filtering, this state estimator alternates prediction and correction. The prediction step uses either a direct image evaluator (IMAGESP) or a procedure for the computation of inverse images based on the algorithm SIVIA. IMAGESP divides the boxes of the original subpaving into smaller boxes and merges the images of all these boxes into an approximate image subpaving. SIVIA proceeds in the opposite direction, starting from the image set, and is much more efficient when applicable. From the predicted subpaving thus obtained, the correction step selects all values of the

state vector that are consistent with the newly available observations. The state estimator has been applied to a simple example, nevertheless representative of some difficulties encountered by classical state estimation algorithms when dealing with hybrid systems.

A key element for a recursive computer implementation of the estimator was the introduction of subpavings, which allow the computation of outer approximations for sets, with any desired precision. Moreover, using IMAGESP and the subpaving class makes it easy to overload the different arithmetic operators and the usual functions on floating-point numbers to get a *subpaving computation* that approximates computation on more general sets.

The main limitation of such techniques lies in the explosion of complexity with the number of state variables. This state estimation technique can nevertheless solve many actual tracking problems. Its application to the tracking of a mobile robot is currently under development.

References

[1] J. Beidler. *Data Structures and Algorithms.* Springer-Verlag, New York, 1996.

[2] M. Berger. *Geometry I and II.* Springer-Verlag, Berlin, 1987.

[3] G. Chen, J. Wang, and L. S. Shieh. Interval Kalman filtering. *IEEE Transaction on Aerospace and Electronic Systems*, 33(1):250–258, 1997.

[4] C. Durieu, B. Polyak, and E. Walter. Ellipsoidal state outer-bounding for mimo systems via analytical techniques. In *IMACS—IEEE-SMC CESA'96 Symposium on Modelling and Simulation*, volume 2, pages 843–848, Lille, 1996.

[5] C. Durieu, B. Polyak, and E. Walter. Trace versus determinant in ellipsoidal outer bounding with application to state estimation. In *Proc. 13th IFAC World Congress*, volume I, pages 43–48, San Francisco, 1996.

[6] R. Hammer, M. Hocks, U. Kulish, and D. Ratz. *C++ Toolbox for Verified Computing.* Springer Verlag, Berlin Heidelberg, 1995.

[7] L. Jaulin. *Solution globale et garantie de problèmes ensemblistes ; application à l'estimation non linéaire et à la commande robuste.* Thèse de doctorat, Université Paris-Sud, Orsay, 1994.

[8] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064, 1993.

[9] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. Internal report (long version of this paper), Laboratoire des Signaux et Systèmes, july 1998.

[10] D. Maksarov and J. P. Norton. State bounding with ellipsoidal set description of the uncertainty. *Int. J. of Control*, 65(5):847–866, 1996.

[11] R. E. Moore. *Methods and Applications of Interval Analysis.* SIAM Publ., Philadelphia, Pennsylvania, 1979.

[12] A. Neumaier. *Interval Methods for Systems of Equations.* Cambridge University Press, London, 1990.

[13] F. C. Schweppe. Recursive state estimation: Unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.

[14] F. C. Schweppe. *Uncertain Dynamic Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

A Some functions manipulating subpavings

The following functions and operators are used in SIVIA. To improve readability, the variables and operators are those used in the text, not always at the C++ standard.

- *union operator* (`spaving& operator ∪ (spaving` $\widehat{\mathcal{X}}$`, spaving` $\widehat{\mathcal{Y}}$`)`): returns the minimal subpaving corresponding to $\widehat{\mathcal{X}} \cup \widehat{\mathcal{Y}}$.
- *box-inclusion test* (`bool operator⊂ (ivector [x], spaving` $\widehat{\mathcal{X}}$`)`): checks whether the interval vector [x] is included in the subpaving $\widehat{\mathcal{X}}$.
- *l-depth expansion function* (`spaving& expand(spaving` $\widehat{\mathcal{X}}$`, int l)`): creates a nonminimal subpaving whose leafs are all at a specified depth l. If the original tree $\widehat{\mathcal{X}}$ has deeper leafs, these leafs are removed, else, any leaf at depth $k < l$, is replaced by a subtree such that all its leafs are at depth $l - k$.
- *subpaving inclusion test* (`bool operator⊂ (spaving` $\widehat{\mathcal{X}}$`, spaving` $\widehat{\mathcal{Y}}$`)`): checks whether the subpaving $\widehat{\mathcal{X}}$ is included in the subpaving $\widehat{\mathcal{Y}}$.

B Recursive Sivia

The main procedure of SIVIA is as follows:

```
spaving& SIVIA(spaving X̂, function_ptr f[],
spaving Ŷ, float ε)
{
  if isempty(X̂) return NULL;
  if disjoint(f[](_box(X̂)),Ŷ) return NULL;
  if (f[](_box(X̂))⊂ Ŷ)||(w(_box(X̂)) < ε)
    return _spaving(_box(X̂));
  if isleaf(X̂) X̂ =expand(X̂,1);
  return ( SIVIA(_lchild(X̂), f[], Ŷ, ε)
    ∪ SIVIA(_rchild(X̂), f[], Ŷ, ε) );
}
```